UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/722,761	11/26/2003	Charles S. Shorb	343355600055	7202

7590 08/22/2007

John V. Biernacki
Jones Day
North Point
901 Lakeside Avenue
Cleveland, OH 44114

EXAMINER	
TSAI, SHENG JEN	

ART UNIT	PAPER NUMBER
2186	

MAIL DATE	DELIVERY MODE
08/22/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	**Application No.** 10/722,761	**Applicant(s)** SHORB, CHARLES S.
	Examiner Sheng-Jen Tsai	**Art Unit** 2186

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 26 June 2007.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-44 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-44 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on 26 November 2003 is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.	6) ☐ Other: _____.

DETAILED ACTION

1. This Office Action is taken in response to Applicant's Amendments and Remarks filed on June 26, 2007 regarding application 10,722,761 filed on November 26, 2003.

2. Claim 44 has been amended.

Claims 1-44 are pending for consideration.

3. ***Response to Remarks and Amendments***

Applicants' amendments and remarks have been fully and carefully considered, with the Examiner's response set forth below.

(1) Applicant contends that the references relied on (Daynes, US 6,343,339 and Ofer, US 6,691,194) can not be combined because Daynes expressly teaches away form such combination and Ofer is limited to hardware processors and does not provide a multiple executing entities such as threads. The Examiner disagrees.

First, Applicant points to column 8, lines 37-42 of Daynes as support of Daynes' teaching away from combination with Ofer. Column 8, lines 37-42 of Daynes recite "As described above, prior art methods provide for the creation of locks in lock data structures. <u>The present invention does not utilize locks with a material existence in the form of lock data structures</u> but <u>provides for immutable lock states</u> (lock states that are not capable of or susceptible to change). Each lock state represents a particular state of one or more locks. <u>Resources share the same immutable lock state if the state of their respective lock is equal</u>. During its lifetime, a resource may be associated with various lock states, each lock state being the representation of the lock of that resource at a given time. Locking operations change the association between a resource and a lock state, should a change of state be necessary. <u>This association materializes the resource's lock</u>. Prior to granting a lock, a determination is made as to whether the lock to be granted conflicts with an existing lock."

Thus, instead of utilizing an individual lock with certain data structure to each resource, Daynes's invention utilizes a plurality of lock states and resources whose locks having the same state will share the lock state. Further, each lock state is implemented using data structure [figure 11], even though the data structure may not be the same as that of "locks with a material existence in the form of lock data structures."

Daynes also teaches that "Each resource is associated with a lock state that represents its lock. Lock states are made of one set of transactions per locking mode. Resources may share the same lock state if the state of their respective locks is equal. <u>Locking operations change the association between a resource and a lock state to reflect changes to the resource's lock</u> (abstract)."

Thus, there is a one-to-one correspondence between a resource's lock and a lock state, and any change to a resource's lock is reflected directly in the associated lock state. Thus, the pitfalls of deadlock or starvation situations described by Ofer [column 2, lines 15-16] would still occur without atomic operation, and the encapsulation of both lock status data and reader data as a single unit for atomic operations disclosed by Ofer to further reduce these pitfalls would be directly applicable to Daynes' scheme, with the understanding that the atomic operation is applied in encapsulation of the data structure associated with the plurality of lock states.

Second, Ofer's invention is not limited to only hardware processors, as Ofer teaches that "Referring now to FIG. 1, computer system 10 is shown to include, among other things, a plurality of processors 1a-1n, running processes A-N, coupled to a shared resource 4 via one or more first common communication channels 3a-n and to a shared memory 2 via one or more second common communication channels 7a-n ... Any or all of processors 1a-1n may request access to shared resource 4 in order to execute their processes A-N. The processors are actual or virtual digital processing units which include one or more CPU's and additional local memory 5a-n (column 6, lines 4-15)."

Thus, the key element of Ofer's invention is the A-N processes, not the processors, because the processors may be virtual units emulated by software. Therefore, the processes of Ofer correspond directly to the transactions of Daynes, and both correspond to the threads recited in the claims.

(2) Applicant contends that there are certain drawbacks of Daynes' method and apparatus, including increasing memory consumption and processing cost if no redundant lock states are present.

However, the presence of a redundant state is not a limitation recited by the claims, thus this argument is irrelevant.

(3) Applicant contends that Daynes' method and apparatus are based on tasks, not threads. The Examiner disagrees.

First, according to the definition by Microsoft Computer Dictionary (5th edition, Microsoft Press, 2002, page 518), a "thread" is "a process that is part of a larger process or program."

Second, Daynes defines a "transaction" as "each time a computer system performs an activity, the activity is referred to as a transaction (column 1, lines 41-42)."

Thus, a transaction is any activity, or process, performed by a computer system, which normally runs complex programs comprising subroutines. Each of these subroutines is considered as an activity, or a process, embedded in the corresponding main routine. Thus, it qualifies as a thread.

(4) Applicant contends that Ofer's method and apparatus does not allow for multiple executable entities' access to be concurrent with respect to a resource. The Examiner disagrees.

First, Applicant agrees that Ofer's invention is directed toward a method and apparatus for improving performance in a system where multiple processors contend for control of a shared resource. The "multiple processors" are the "multiple executable entities" recited in claim 44.

Second, the multiple processors or processes all attempt to access, at least, the shared lock concurrently to determine whether if a target resource is available before accessing the desired target. Note that the shared lock is a also resource by itself.

Third, it is noted that Daynes also teaches the amended limitation because figure 6, TIME T1 shows that transactions T1 and T2 are concurrently reading form resources O1 and O2.

Thus both Daynes and Ofer teach the amended limitation of claim 44.

(5) Therefore, the Examiner's position regarding the patentability of all claims remains the same as stated in the previous Office Action.

Claim Rejections - 35 USC § 112

4. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

5. Claim 44 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 44 recites the limitation of "wherein the multiple executable entities' access is <u>a concurrent access</u> to the at least one resource." It is not clear what constitutes a "concurrent access."

First, The American Heritage College Dictionary defines the term "concurrent" as "happening at the same time." The question, then, is what is considered as "at the same time." Is it within a window of 1 nano second? Or is it 1 mille second? Or is it 1 second? Or is it within a window that may cover all the activities of multiple operations?

Second, it is well known that only one write operation may occur at a time because of its nature to alter the content. It is conceivable that multiple read operations may all occur "at the same time," but the underlying assumption is that the physical resource has multiple input/output ports, at least one for each read operation, so that all of them may happen at the same time. However, the Specification of the Application is completely silent on this aspect, and it is not clear if this is Applicant's intension.

Third, as an analogy, considering the scenario where customers are doing transactions in a bank. Some customers are being served by tellers and the others are waiting in a line. Are all customers being concurrently served by tellers? They are not. But are all of them concurrently in the process of doing their transactions in a bank? They are. Thus the answer seems to depend on which resource, the bank or the teller.

Fourth, contention of a resource occurs only when multiple requests are directed to the same resource simultaneously, or concurrently, at least during the time window defined by how long it takes the resource to serve a request. If there is no contention, then a lock would not be activated. Thus the fact that a lock has been activated implies the presence of multiple requests being directed to the same resource simultaneously, or concurrently.

In the subsequent claim analysis, the Examiner interprets the limitation "wherein the multiple executable entities' access is a concurrent access to the at least one resource" as "more than one of the multiple executable entities access the shared lock during the same time window which starts at the instant of the earliest request and ends at the instant that all requests are completed."

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1, 3-27 and 32-44 are rejected under 35 U.S.C. 103(a) as being unpatentable over Daynes (U.S. 6,343,339), and in view of Ofer (US 6,691,194).

As to claim 1, the references discloses **a memory for storing a computer-implemented shared locking data store** [Daynes: A lock is comprised of a data structure that records the identity of the transactions that are given the right to execute operations on the resource the lock protects. Each time a lock is acquired, memory is used for the lock data structure (column 3, lines 31-39); Ofer: figures 3-4; a main lock data structure is implemented in a shared memory accessible ... (column 3, lines 44-52)] **for handling multiple executable entities' access to at least one resource** [Daynes: Each time a computer system performs an activity, the activity is referred to as a transaction. ... When executing transactions, a computer system may execute a group of transactions at one time (column 1, lines 40-62); One type of lock is a shared lock. A shared lock permits multiple transactions to read (view) an item simultaneously without any modification or addition to the item (no writing is permitted) (column 2, lines 18-21); Ofer: figure 2; column 3, lines 44-52], **said shared locking data store being stored in a computer memory** [Daynes: A lock is comprised of a data structure that records the identity of the transactions that are given the right to execute operations on the resource the lock protects. Each time a lock is acquired, memory is used for the lock data structure (column 3, lines 31-39); Ofer: figures 3-4; a main lock data structure is implemented in a shared memory accessible ... (column 3, lines 44-52)], **said shared locking data store comprising**:

write requested data that is indicative of when a write request for the resource has been made [Daynes: figures 3-4, write lock (W); figure 7, step 702, "any pending lock requests or conflicts?" step 704C, "place lock request in queue;" A lock state is comprised of a set of transactions that own a lock in a specific mode. Among other modes, a locking mode may comprise a read mode or a write mode (column 4, lines 2-5); Additionally, locks must be administered to provide a queue for transactions that are waiting to acquire a lock, and to rollback any executed actions if a deadlock results (i.e., when each of two transactions are waiting for a lock release from the other before continuing) (column 2, lines 29-41);

Ofer: column 9, lines 43-67, column 10, lines 1-67 and column 11, lines 1-31; although Ofer does not explicitly specify a request to be a read or a write, it is understood that any request must be either a read or a write];

writer active data that is indicative of whether a writer process is actively undergoing a write operation with respect to the resource [Daynes: single write owner (SRO): the lock state type that represents ownership by a single owner in write mode. There is one lock state of this type per active transaction (column 16, lines 10-13); figures 3-4; figure 7; conflict detection, column 18, lines 7-14;

Ofer: column 9, lines 43-67, column 10, lines 1-67 and column 11, lines 1-31; although Ofer does not explicitly specify a request to be a read or a write, it is understood that any request must be either a read or a write];

wait event posted data, wherein the wait event posted data is indicative of whether a read request to the resource has completed, wherein the wait event

posted data is used to indicate when a write request can access the resource [Daynes: Additionally, locks must be administered to provide a queue for transactions that are waiting to acquire a lock, and to rollback any executed actions if a deadlock results (i.e., when each of two transactions are waiting for a lock release from the other before continuing) (column 2, lines 29-41); In one embodiment, the queue is processed in first-in-first-out (FIFO) order. At step 706, the lock manager waits for the requested lock to become available. In one embodiment, a lock may become available when the conflict is cleared (e.g., when the lock is released by another transaction) and when transactions that were ahead of the current requestor in the queue have been processed (column 11, lines 26-33);
Ofer: the queue lock is implemented by a main lock data structure (column 6, lines 38-41)];

reader data that is indicative of whether a reader process is active and attempting to read from the resource [Daynes: figures 3-4, read lock (R); figure 7, step 702, "any pending lock requests or conflicts?" step 704C, "place lock request in queue;" A lock state is comprised of a set of transactions that own a lock in a specific mode. Among other modes, a locking mode may comprise a read mode or a write mode (column 4, lines 2-5); Additionally, locks must be administered to provide a queue for transactions that are waiting to acquire a lock, and to rollback any executed actions if a deadlock results (i.e., when each of two transactions are waiting for a lock release from the other before continuing) (column 2, lines 29-41);

Ofer: column 9, lines 43-67, column 10, lines 1-67 and column 11, lines 1-31; although Ofer does not explicitly specify a request to be a read or a write, it is understood that any request must be either a read or a write];

wherein the shared locking data store allows writer and reader locking state information to be determined so that access to the resource can be handled [Daynes: figures 4-7; figure 11; Method and apparatus for locking by sharing lock states. Each resource is associated with a lock state that represents its lock. Lock states are made of one set of transactions per locking mode (abstract);

Ofer: figures 5-9; column 9, lines 43-67, column 10, lines 1-67 and column 11, lines 1-31];

wherein encapsulation of both lock status data and the reader data in the shared locking data store allows a hardware atomic operation to operate upon both the lock status data and the reader data as a single unit for determining how access to the resource is to be handled [Daynes: In one embodiment, lock state sharing enables the usage of non-blocking synchronizations to change the lock state of a resource. A non-blocking synchronization requires an implementation using an atomic compare and swap operation (such as the cas instruction of the Sparc V9 processor, or the cmpxchg instruction of the Intel486 and Pentium family of processors) (column 17, lines 18-56);

Ofer: the main lock data structure provides, in a single atomic structure, the resources needed to lock a shared resource (column 3, lines 47-49); column 4, lines 51-56; column 5, lines 8-14].

Regarding claim 1, Daynes does not teach the limitation of "**encapsulation of both lock status data and the reader data in the shared locking data store allows a hardware atomic operation to operate upon both the lock status data and the reader data as a single unit.**"

However, Daynes does teach using an atomic compare and swap operation for determining how access to the resource should be handled [column 17, lines 18-56].

Further, Ofer discloses in the invention "Selective Association of Lock Override Procedures with Queued Multimode Lock" a lock data structure [figures 3-4; column 3, lines 44-52] which **encapsulate both lock status data** [LOCK_PW, figure 3, 35; column10, lines 28-34] **and the reader data** [NEXT_FREE, figure 3, 39; column 10, lines 40-44] **in the shared locking data store** [figures 3-4; column 3, lines 44-52] **allows a hardware atomic operation to operate upon both the lock status data and the reader data as a single unit** [the main lock data structure provides, in a single atomic structure, the resources needed to lock a shared resource (column 3, lines 47-49); column 4, lines 51-56; column 5, lines 8-14] **for determining how access to the resource is to be handled** [figures 5-9].

Encapsulation of both lock status data and reader data as a single unit for atomic operations further reduces the pitfalls of deadlock or starvation situations [Ofer, column 2, lines 15-16].

Therefore, it would have been obvious for one of ordinary skills in the art at the time of Applicant's invention to recognize that benefits of encapsulation of both lock status data and reader data as a single unit for atomic operations, as demonstrated by

Ofer, and to incorporate it into the existing scheme disclosed by Daynes to further reduce the pitfalls of deadlock or starvation situations.

As to claim 3, Daynes teaches that **the multiple executable entities include threads which are to access the resource** [Each time a computer system performs an activity, the activity is referred to as a <u>transaction</u>. ... When executing transactions, a computer system may execute <u>a group of transactions</u> at one time (column 1, lines 40-62); One type of lock is a shared lock. A shared lock permits <u>multiple transactions</u> to read (view) an item simultaneously without any modification or addition to the item (no writing is permitted) (column 2, lines 18-21)], **wherein the threads comprise a writer thread and a reader thread** [figures 3-4; read (R) and write (W)], **wherein the locking state information is used to determine how the writer thread and the reader thread access the resource** [figures 4-7; figure 11; Method and apparatus for locking by sharing lock states. Each resource is associated with a lock state that represents its lock. Lock states are made of one set of transactions per locking mode (abstract)].

As to claim 4, Daynes teaches that **the resource requires protection on a per thread basis** [figures 3-4, the lock state of read (R) and write (W) is based on a per transaction (T) basis].

As to claim 5, Daynes teaches that **the locking state information indicates where in the overall locking process an operating system is with respect to the resource** [figures 4-7; figure 11].

As to claim 6, Daynes teaches that **the request of the reader thread is allowed to succeed unless there is a write request active or pending as indicated by the write requested data and the writer active data** [figure 7].

As to claim 7, Daynes teaches that **rules mechanism that indicates how resource access requests are to be handled based upon the data stored in the shared locking data store** [figures 4-7; figure 11; Method and apparatus for locking by sharing lock states. Each resource is associated with a lock state that represents its lock. Lock states are made of one set of transactions per locking mode (abstract); Additionally, locks must be administered to provide a queue for transactions that are waiting to acquire a lock, and to rollback any executed actions if a deadlock results (i.e., when each of two transactions are waiting for a lock release from the other before continuing) (column 2, lines 29-41)].

As to claim 8, Daynes teaches that **the rules mechanism includes allowing any number of read requests to succeed unless there is a writer active or pending as indicated by the shared locking data store** [multiple read owner (MRO): the lock state type that represents ownership by multiple owners in read mode only (column 16, lines 14-16); figure 7; conflict detection, column 18, lines 7-14].

As to claim 9, Daynes teaches that **the rules mechanism includes allowing only one writer to be active at any given time** [single write owner (SRO): the lock state type that represents ownership by a single owner in write mode. There is one lock state of this type per active transaction (column 16, lines 10-13); figures 3-4; figure 7; conflict detection, column 18, lines 7-14].

As to claim 10, Daynes teaches that **the rules mechanism includes that no preference is given when deciding which of waiting read or write requests should be honored first** [the procedures shown in figures 3-4 and 7 do not give preference to either read or write requests]

As to claim 11, Daynes teaches that **the rules mechanism substantially eliminates a reader request starvation or a writer request starvation situation with respect to the resource** [the procedures shown in figures 3-4 and 7 do not give preference to either read or write requests, is fair to both read and write requests, hence prevent starvation of either read requests or write requests].

As to claim 12, Daynes teaches **the resource comprises data stored in computer memory** [A transaction may need access to <u>a record in a</u> <u>database</u> or a portion of information in a database (column 1, lines 47-49)].

As to claim 13, Daynes teaches that **the resource comprises a computer file** [A transaction may need access to a record in a database or a portion of information in a database (column 1, lines 47-49)].

As to claim 14, Daynes teaches that **the resource comprises an input/output device** [I/O devices, figure 1, 119].

As to claim 15, Daynes teaches that **the write requested data is to have a set status when a write request has been made** [lock set, column 15, lines 7-39].

As to claim 16, Daynes teaches that **a set status for the write requested data indicates whether an operating system (OS) mutex lock is to be used for processing access to the resource** [Additionally, locks must be administered to provide a queue for transactions that are waiting to acquire a lock, and to rollback any executed actions if a deadlock results (i.e., when each of two transactions are waiting for a lock release from the other before continuing) (column 2, lines 29-41); In one embodiment, the queue is processed in first-in-first-out (FIFO) order. At step 706, the lock manager waits for the requested lock to become available. In one embodiment, a lock may become available when the conflict is cleared (e.g., when the lock is released by another transaction) and when transactions that were ahead of the current requestor in the queue have been processed (column 11, lines 26-33)].

As to claim 17, Daynes teaches that **indication of whether a writer process is active by the writer active data is used to protect against readers posting an operating system (OS) event after the writer thread has been activated** [Additionally, locks must be administered to provide a queue for transactions that are waiting to acquire a lock, and to rollback any executed actions if a deadlock results (i.e., when each of two transactions are waiting for a lock release from the other before continuing) (column 2, lines 29-41); In one embodiment, the queue is processed in first-in-first-out (FIFO) order. At step 706, the lock manager waits for the requested lock to become available. In one embodiment, a lock may become available when the conflict is cleared (e.g., when the lock is released by another transaction) and when transactions that were ahead of the current requestor in the queue have been processed (column 11, lines 26-33)].

As to claim 18, Daynes teaches that **the shared locking data store of claim 1 further comprising wait event posted data, wherein the wait event posted data is**

indicative of whether a read request to the resource has completed, wherein the wait event posted data is used to indicate when a write request can access the resource, wherein a last active read clears status of the wait event posted data prior to posting an event [Additionally, locks must be administered to provide a queue for transactions that are waiting to acquire a lock, and to rollback any executed actions if a deadlock results (i.e., when each of two transactions are waiting for a lock release from the other before continuing) (column 2, lines 29-41); In one embodiment, the queue is processed in first-in-first-out (FIFO) order. At step 706, the lock manager waits for the requested lock to become available. In one embodiment, a lock may become available when the conflict is cleared (e.g., when the lock is released by another transaction) and when transactions that were ahead of the current requestor in the queue have been processed (column 11, lines 26-33)].

As to claim 19, Daynes teaches that **the clearing ensures that one and only one posting to a writer of an event occurs** [In one embodiment, a lock may become available when the conflict is cleared (e.g., when the lock is released by another transaction) and when transactions that were ahead of the current requestor in the queue have been processed (column 11, lines 26-33)].

As to claim 20, Daynes teaches that **the posting occurs if status of the writer active data is not set** [figure 7, step 708, "value of new lock state computed" as a result of no pending lock requests or conflicts (step 702)].

As to claim 21, Daynes teaches that **the processes comprise threads, wherein the wait event posted data indicates when events can be posted by one thread to**

notify another thread that the other thread's request can be processed [figure 10 shows that the read (R) and Write (W) tables indicate which transactions may be using which resources].

As to claim 22, Daynes teaches that **the wait event posted data indicates whether a reader thread can post a lock release event to a writer thread** [figure 10].

As to claim 23, Daynes teaches that **the wait event posted data is used to prevent multiple reader threads from posting redundant lock release events** [figure 10; column 13, lines 61-67, column 14, lines 1-18].

As to claim 24, Daynes teaches that **the write requested data, writer active data, and wait event posted data provide multi-threaded protection in place of an operating system (OS) mutex** [One type of lock is a shared lock. A shared lock permits multiple transactions to read (view) an item simultaneously without any modification or addition to the item (no writing is permitted) (column 2, lines 18-21); Additionally, locks must be administered to provide a queue for transactions that are waiting to acquire a lock, and to rollback any executed actions if a deadlock results (i.e., when each of two transactions are waiting for a lock release from the other before continuing) (column 2, lines 29-41); In one embodiment, the queue is processed in first-in-first-out (FIFO) order. At step 706, the lock manager waits for the requested lock to become available. In one embodiment, a lock may become available when the conflict is cleared (e.g., when the lock is released by another transaction) and when transactions that were ahead of the current requestor in the queue have been processed (column 11, lines 26-33].

As to claim 25, Daynes teaches that **the shared locking data store of claim 18, wherein locking state data comprises the write requested data, the writer active data, the reader count data, and the wait event posted data** [refer to "As to claim 1"]; wherein the locking state data is formatted as a single atomic unit [SRO (figure 11, 1120) and MRO (figure 11, 1126); column17, lines 17-56].

As to claim 26, Daynes teaches that **the shared locking data store of claim 25, wherein the locking state data is formatted as a multi-bit single unit** [SRO (figure 11, 1120) and MRO (figure 11, 1126); In one embodiment of the invention, lock owner sets are represented as bitmaps. A bitmap is an array of binary digits (either a 1 or 0 in the binary number system; also referred to as bits). Additionally, each transaction is assigned a locking context that uniquely identifies the transaction (column 14, lines 19-30)].

As to claim 27, Daynes teaches that **the shared locking data store of claim 26, wherein the formatting of the locking state data as a single unit allows an operating system to use atomic operations upon the locking state data** [In one embodiment, lock state sharing enables the usage of non-blocking synchronizations to change the lock state of a resource. A non-blocking synchronization requires an implementation using <u>an atomic compare and swap operation</u> (such as the cas instruction of the Sparc V9 processor, or the cmpxchg instruction of the Intel486 and Pentium family of processors) (column 17, lines 18-56)].

As to claim 32, Daynes teaches that **the locking state data is formatted as a multi-bit single unit** [SRO (figure 11, 1120) and MRO (figure 11, 1126); In one embodiment of the invention, lock owner sets are represented as bitmaps. A bitmap is an array of binary digits (either a 1 or 0 in the binary number system; also referred to as bits). Additionally, each transaction is assigned a locking context that uniquely identifies the transaction (column 14, lines 19-30)];

wherein the write requested data comprises a single bit in the unit [figure 11, column 14, lines 19-30];

wherein the writer active data comprises a single bit in the unit [figure 11, column 14, lines 19-30];

wherein the wait event posted data comprises a single bit in the unit [figure 11, column 14, lines 19-30];

wherein the reader data comprises a plurality of bits in the unit to indicate number of active readers with respect to the resource [figure 11, column 14, lines 19-30].

As to claim 33, refer to "As to claim 32."

As to claim 34, Daynes teaches that **the reader data's count of the readers provides an indication as to whether readers are present and when the last reader exits** [figure 11, column 14, lines 19-30; figures 3-4 and 10].

As to claim 35, Daynes teaches that **a read lock acquisition means accesses the locking state data for processing a read lock acquisition of the resource** [figures 3-4, read lock (R); figure 7, step 702, "any pending lock requests or conflicts?"

step 704C, “place lock request in queue;” A lock state is comprised of a set of transactions that own a lock in a specific mode. Among other modes, a locking mode may comprise a read mode or a write mode (column 4, lines 2-5)].

As to claim 36, Daynes teaches that **a read lock release means accesses the locking state data for processing a read lock release of the resource** [figure 10 shows how to acquire and release a read lock].

As to claim 37, Daynes teaches that **a write lock acquisition means accesses the locking state data for processing a write lock acquisition of the resource** [figures 3-4, write lock (W); figure 7, step 702, “any pending lock requests or conflicts?” step 704C, “place lock request in queue;” A lock state is comprised of a set of transactions that own a lock in a specific mode. Among other modes, a locking mode may comprise a read mode or a write mode (column 4, lines 2-5)].

As to claim 38, Daynes teaches that **a write lock release means accesses the locking state data for processing a write lock release of the resource** [To release a lock for a specific resource, the transaction determines the lock state value that will result after removing itself from the lock state for that resource (column 4, lines 28-30)].

As to claim 39, Daynes teaches that **the operating system returns a lock busy status indicator when a lock cannot be obtained within a predetermined time** [Additionally, locks must be administered to provide a queue for transactions that are waiting to acquire a lock, and to rollback any executed actions if a deadlock results (i.e., when each of two transactions are waiting for a lock release from the other before continuing) (column 2, lines 29-41); In one embodiment, the queue is processed in first-in-first-out (FIFO) order. At step 706, the lock manager waits for the requested lock to become available. In one embodiment, a lock may become available when the conflict is cleared (e.g., when the lock is released by another transaction) and when transactions that were ahead of the current requestor in the queue have been processed (column 11, lines 26-33].

As to claim 40, refer to "As to claim 1" presented earlier in this Office Action.

As to claim 41, refer to "As to claim 1" presented earlier in this Office Action.

As to claim 42, refer to "As to claim 1" presented earlier in this Office Action.

As to claim 43, refer to "As to claim 1" presented earlier in this Office Action.

As to claim 44, refer to "As to claim 1" presented earlier in this Office Action.
Further, both Daynes and Ofer teach the multiple executable entities's access is a concurrent access to at least one resource [Daynes: figure 6, TIME T1 shows that transactions T1 and T2 are concurrently reading form resources O1 and O2; Ofer: the invention is directed toward a method and apparatus for improving performance in a system where <u>multiple processors</u> contend for control of a shared resource. The "multiple processors" are the "multiple executable entities" recited in claim 44. The multiple processors or processes all attempt to access, at least, the shared lock concurrently to determine whether if a target resource is available before accessing the desired target. Note that the shared lock is a also resource by itself].

8. Claim 2 is rejected under 35 U.S.C. 103(a) as being unpatentable over Daynes (U.S. 6,343,339), and in view of Ofer (US 6,691,194).

As to claim 2, Daynes teaches that **the executable entities include processes, threads, stand-alone application or code that runs at the request of another program wherein use of the shared locking data stores allows for the avoidance of an operating system call or avoidance of a resource accessing trap of the operating system's kernel** [refer to "As to claim 1"].

Regarding claim 2, neither Daynes nor Ofer mention that **the executable entities include daemons, applets, servlets or ActiveX components**.

However, these entities are well known in the art and have been widely used in computer systems (see Microsoft Computer Dictionary, 5th edition, Microsoft Press, 2002, isbn 0-7356-1495-4; page 140 – daemon; page 31 – applet; page 18 – ActiveX; page 475 – servlet).

Therefore, it would have been obvious for one of ordinary skills in the art at the time of Applicant's invention to recognize that these entities are well known in the art and have been widely used in computer systems, hence lacking patentable significance.

9. Claims 28-31 are rejected under 35 U.S.C. 103(a) as being unpatentable over Daynes (U.S. 6,343,339), in view of Ofer (US 6,691,194), and further in view of Mckenney et al. (US 6,480,918).

As to claims 28-31, Daynes teaches the use of **atomic sub** operation [A non-blocking synchronization requires an implementation using <u>an atomic compare and swap operation</u> (such as the cas instruction of the Sparc V9 processor, or the cmpxchg instruction of the Intel486 and Pentium family of processors) (column 17, lines 18-56)], but does not mention that the atomic operations include **atomic get, atomic set, and atomic add operation**.

However, these atomic operations are well known and commonly used in the art.

Further, Mckenney et al. disclose in their invention "Lingering Locks with Fairness Control for Multi-Node Computer Systems" a scheme of lock management in a multiprocessor computer system [abstract] wherein atomic get [atomically acquiring the lock (column 24, lines 65-66)], atomic set [atomically setting the available bit (column 25, lines 25-26)], atomic add [atomic add (atomic_xadd_uint) (column 10, lines 29-30)] and atomic sub [atomic compare-and-exchange (atomic_cmp_xchng_uint) (column 10, lines 30-31)] operations are used to facilitate the control of locks.

Therefore, it would have been obvious for one of ordinary skills in the art at the time of Applicant's invention to recognize that these atomic operations are well known in the art and have been widely used in computer systems, as demonstrated by Mckenney et al., hence lacking patentable significance.

10. *Related Prior Art of Record*

The following list of prior art is considered to be pertinent to applicant's invention, but not relied upon for claim analysis conducted above.

- Oliver, (US 6,029190), "Read Lock and Write Lock management system Based upon MUTEX and Semaphore Availability."
- Ofer, (US 6,718,448), "Queued Locking of a Shared Resource Using Multimodal Lock Types."

- McClaughry et al., (US 5,933,825), "Arbitrating Concurrent Access to File system Objects." ✓
- McKenney, (US Patent Application Publication 2004/0117531), "Adaptive Reader-Writer Lock."
- Kakivaya et al., (US 6,546,443), "Concurrency-Safe Reader-Writer Lock with Time Out Support."
- Clark, (US 6,598,068), "Method and Apparatus for Automatically Managing Concurrent Access to a Shared Resource in a Multi-Threaded Programming Environment."
- Bacon, (US 6,247,025), "Locking and Unlocking Mechanism for Controlling Concurrent Access to Objects."
- Watanabe et al., (US 6,016,490), "Database Management System."
- Harris, (US Patent Application Publication 2003/0200398), "Method and Apparatus for Emulating Shared Memory in a Storage Controller."
- Onodera et al., (US 6,883,026), "Method and Apparatus for Managing Locks of Objects and Method and Apparatus for Unlocking Objects."
- Won et al., (US Patent Application Publication 2003/0126187), "Method and Apparatus for Synchronization in a Multi-Thread system of JAVA Virtual machine."
- Li, (US Patent Application Publication 2004/0059733), "Method and Apparatus for Locking Objects in a Multi-Threaded Environment."

- Li, (US Patent Application Publication 2003/0233393), "Thread Optimization for Lock and Unlock operations in a Multi-Threaded Environment."

Conclusion

11. Claims 1-44 are rejected as explained above.

12. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

13. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Sheng-Jen Tsai whose telephone number is 571-272-4244. The examiner can normally be reached on 8:30 - 5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Matthew Kim can be reached on 571-272-4182. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Sheng-Jen Tsai
Examiner
Art Unit 2186

August 16, 2007

[illegible]

MATTHEW KIM
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100